ซอฟต์แวร์
นริศ วิชาธีรสกุล

วินโดวส์ โปรแกรมมิง : INPUT PART

ติดตามกันต่อกับส่วนของอินพุต ในตอนที่เหลืออยู่... เพื่อประสิทธิภาพของการเขียนโปรแกรม และการสร้างสรรค์งานที่สมบูรณ์แบบ ให้คุณได้ใช้งานวินโดวส์... ได้อย่างเต็มคุณภาพ

สืบเนื่องจากฉบับที่ 84 ได้กล่าวถึงการโปรแกรมมิงสำหรับส่วนอินพุต (Input part) โดยแบ่งใหญ่ ๆ 3 ส่วนคือ ส่วนอินพุตจากแป้นพิมพ์, ส่วนอินพุตจากเม้าส์ และส่วนอินพุตจากเวลา (Timer)

เม้าส์อินพุต

เมื่อเราเข้าสู่ภายใต้วินโดวส์ ไดรเวอร์ต่าง ๆ จะถูกโหลดเข้ามา รวมทั้งเม้าส์ไดรเวอร์ด้วย หน้าที่หลักของเม้าส์ไดรเวอร์คือ รายงานสถานะต่าง ๆ ที่เกิดขึ้นกับเม้าส์โดยออกมาในรูปของเมสเสจ โดยส่งผ่านไปเก็บไว้ในเมสเสจคิว เพื่อรอการดึงไปใช้ ดังนั้นโดยหลักการแล้วถ้าพูดถึงการโปรแกรมมิงที่เกี่ยวกับเม้าส์ภายใต้วินโดวส์คือ การเขียนส่วนโปรแกรมที่ต้องการภายใต้เมสเสจที่เกี่ยวข้องกับเม้าส์ ดังรูปที่ 1

สำหรับเมสเสจที่เกิดเนื่องจากเม้าส์เราสามารถแบ่งได้ดังนี้คือ เมสเสจของเม้าส์ภายใต้พื้นที่ใช้งานของหน้าต่าง (Client Area Mouse Message) และเมสเสจของเม้าส์ที่เกิดนอกพื้นที่ใช้งานของหน้าต่าง (Non-Client Area Mouse Message)

เมื่อสร้างหน้าต่างขึ้นมาดังรูปที่ 2 พื้นที่ใช้งาน (Client Area) คือพื้นที่ที่หน้าต่างหนึ่ง ๆ มีไว้ให้เราใช้ในการแสดงผลโดยผ่าน DC (ดูรายละเอียดเกี่ยวกับ DC ได้ในตอน Painting ในฉบับที่ 82) แต่สำหรับพื้นที่ของส่วนประกอบอื่น ๆ เช่น ส่วนหัว (Caption), ส่วนกรอบ (Frame), ส่วนเมนูระบบของหน้าต่าง (System Menu) และอื่น ๆ นอกเหนือจากไคล์เอ็นต์แอเรีย เราเรียกพื้นที่ของส่วนประกอบพวกนี้ว่า นอนไคล์เอ็นต์แอเรีย (Non-Client Area)

ก่อนที่จะลงลึกในส่วนเมสเสจนั้น มาทำความรู้จักกับศัพท์ที่เกี่ยวข้องกับเม้าส์ก่อนดังนี้

เคอร์เซอร์ (Cursor) คือ รูปภาพหรือสัญลักษณ์ ที่อยู่บนจอภาพที่เราใช้แทนเม้าส์

คลิ๊ก (Click) คือ การกดปุ่มและปล่อยปุ่มของเม้าส์อย่างต่อเนื่องกัน

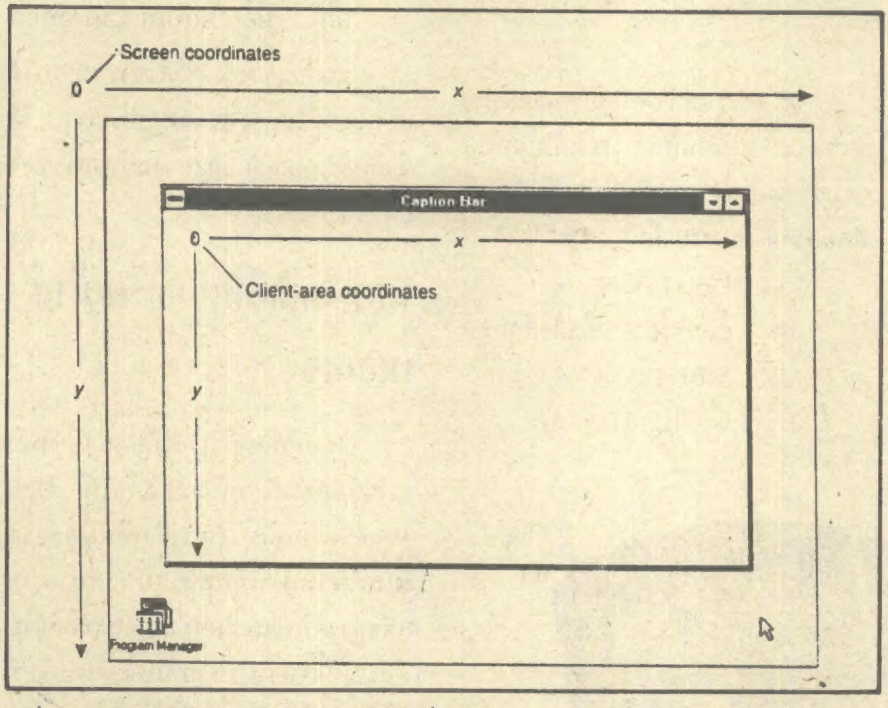
ดับเบิ้ลคลิ๊ก (Double Click) คือ การกดปุ่มและปล่อยปุ่มของเม้าส์อย่างต่อเนื่องกัน 2 จังหวะ

ไคล์เอ็นต์แอเรียเม้าส์เมสเสจ

เมสเสจที่เกิดจากการที่เราเลื่อนเม้าส์หรือทำอะไรกับเม้าส์ในไคล์เอ็นต์แอเรีย ประกอบไปด้วยเมสเสจ 10 เมสเสจ สามารถแบ่งเป็นเมสเสจที่บอกสถานภาพการกดหรือปล่อยปุ่มของเม้าส์ได้ 9 เมสเสจ และ 1 เมสเสจ

```
int WndProc (HWND) hWnd, WORD message, WORD wParam, LONG lParam)
{
...
switch (message)
{
...
case WM_MOUSEMOVE:
        ส่วนโปรแกรมที่ทำงานตามที่เราต้องการเมื่อเกิดการเลื่อนเม้าส์
        ...
case WM_LBUTTONDOWN:
        ส่วนโปรแกรมที่ทำงานตามที่ต้องการเมื่อเกิดการกดปุ่มซ้ายของเม้าส์
        ...
}
...
}
```

รูปที่ 1 แสดงการดักเมสเสจที่เกี่ยวข้องกับเม้าส์ภายใต้วินโดวส์โพรซีเดอร์



รูปที่ 2 แสดงส่วนของไคล์เอ็นต์แอเรียและโคออร์ดิเนตที่อ้างอิง

สำหรับสถานภาพการเลื่อนของเม้าส์

WM_LBUTTONDOWN
เมสเสจแสดงถึงการกดปุ่มซ้ายของเม้าส์

WM_LBUTTONUP
เมสเสจแสดงถึงการปล่อยปุ่มซ้ายของเม้าส์

WM_LBUTTONDBLCLK
เมสเสจแสดงถึงการเกิดดับเบิ้ลคลิ๊กบนปุ่มซ้ายของเม้าส์

WM_MBUTTONDOWN
เมสเสจแสดงถึงการกดปุ่มกลางของเม้าส์ (กรณีเม้าส์ 3 ปุ่ม)

WM_MBUTTONUP
เมสเสจแสดงถึงการปล่อยปุ่มกลางของเม้าส์ (กรณีเม้าส์ 3 ปุ่ม)

WM_MBUTTONDBLCLK
เมสเสจแสดงถึงการเกิดดับเบิ้ลคลิ๊กบนปุ่มกลางของเม้าส์ (กรณีเม้าส์ 3 ปุ่ม)

WM_RBUTTONDOWN
เมสเสจแสดงถึงการกดปุ่มขวาของเม้าส์

WM_RBUTTONUP
เมสเสจแสดงถึงการปล่อยปุ่มขวาของเม้าส์

WM_RBUTTONBLCLK
เมสเสจแสดงถึงการเกิดดับเบิ้ลคลิ๊กบนปุ่มขวาของเม้าส์

WM_MOUSEMOVE
เมสเสจแสดงถึงการเคลื่อนเม้าส์

วินโดว์โพรซีเดอร์ (Window Procedure) เป็นส่วนที่เราเขียนเพื่อจะจัดการกับเมสเสจต่าง ๆ จะสามารถรับเมสเสจ WM_MBUTTON... ก็ต่อเมื่อเม้าส์ที่ใช้เป็นประเภท 3 ปุ่มเท่านั้นและจะได้รับเมสเสจ WM_RBUTTON... ก็ต่อเมื่อเม้าส์ที่ใช้เป็นประเภท 2 หรือ 3 ปุ่ม รวมไปถึงเมสเสจที่แสดงการเกิดดับเบิ้ลคลิ๊ก จะมีก็ต่อเมื่อในส่วนลงทะเบียนของหน้าต่างนั้นมีการกำหนดรูปแบบหน้าต่างเป็น CS_DBLCKS ด้วย ตัวอย่างเช่นกรณีที่เรากำหนดคลาสในส่วนลงทะเบียนดังนี้

```
wndclass.style = CS_HREDRAW
: CS_VREDRAW : CS_DBLCLKS
```

นั่นหมายความว่า กรณีที่ดับเบิ้ลคลิ๊กบนปุ่มซ้ายเราก็จะได้เมสเสจ WM__LBUTTONBLCLK ออกมาเพียงเมสเสจเดียวซึ่งมิฉะนั้นแล้วเราจะได้เมสเสจออกมาถึง 4 เมสเสจ คือ WM__LBUTTONDOWN, WM__LBUTTONUP, WM__LBUTTONDOWN, WM__LBUTTONUP ตามลำดับ

สำหรับ lParam (Long Parameter) ที่เกี่ยวข้องกับเมสเสจข้างต้นทั้งหมดจะบรรจุตำแหน่งของเม้าส์ในขณะนั้น โดยที่ตำแหน่งในแนวแกน x จะอยู่ที่เวิร์ดต่ำ (สองไบต์ต่ำ) และตำแหน่งของเม้าส์ในแกน y จะอยู่ที่เวิร์ดสูง (สองไบต์สูง)

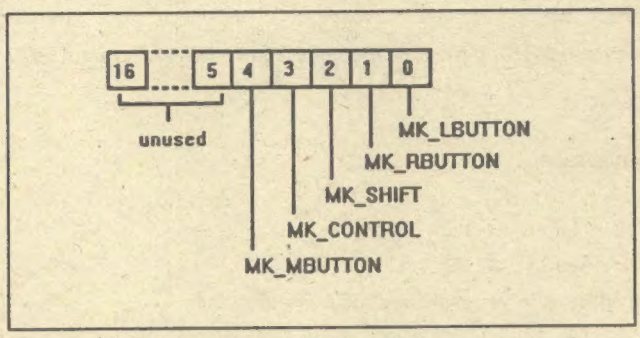
ค่าใน lParam สามารถนำมาแยกใช้ในรูปเวิร์ดได้โดยอาศัย แมคโคร LOWORD หรือ HIWORD (กำหนดใน WINDOWS.H) ทั้งนี้เราจะใช้ LOWORD เมื่อเราต้องการค่าของเวิร์ดต่ำ และใช้ HIWORD เมื่อเราต้องการค่าของเวิร์ดสูง ตัวอย่างเช่น

```
wXpos = LOWORD(lParam);/* กำหนดค่าเวิร์ดต่ำให้กับตัวแปร wXpos */
wYpos = HIWORD(lParam);/* กำหนดค่าเวิร์ดสูงให้กับตัวแปร wYpos*/
```

ดูที่ wParam (Word Parameter) ขนาด 2 ไบต์ที่มีถูกกำหนดเป็นแฟลกเพื่อใช้บอกสถานะเกี่ยวกับเม้าส์ซึ่งมีด้วยกัน 5 สถานะดังนี้ (ดูรูปที่ 3)

MK__LBUTTON
เป็น 1 เมื่อปุ่มซ้ายของเม้าส์ถูกกด

MK__MBUTTON
เป็น 1 เมื่อปุ่มกลางถูกกด



รูปที่ 3 แสดงแฟลกที่บอกสถานะของเม้าส์ในไคล์เอ็นแอเรีย

MK__RBUTTON
เป็น 1 เมื่อปุ่มขวาถูกกด

MK__SHIFT
เป็น 1 เมื่อแป้นพิมพ์ shift ถูกกด

MK__CONTROL
เป็น 1 เมื่อแป้นพิมพ์ Ctrl ถูกกด

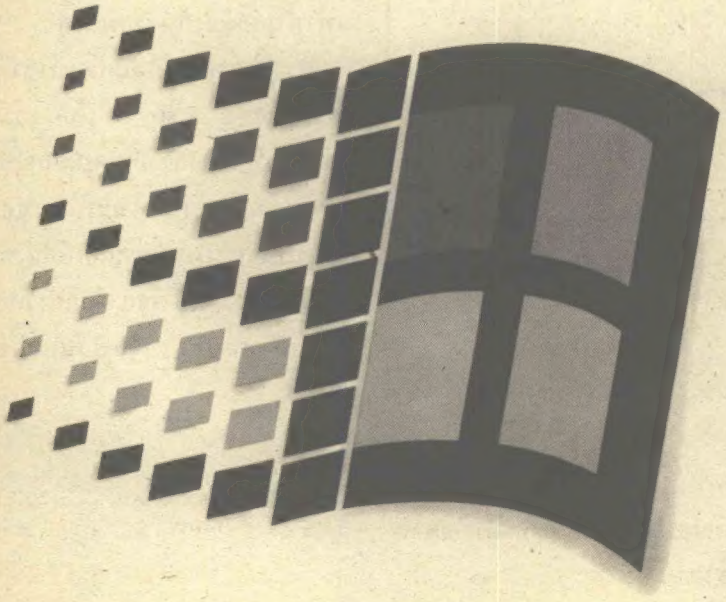
สถานะข้างต้นเราสามารถนำมาใช้ประโยชน์ในการตรวจสอบต่าง ๆ ได้ ซึ่งก็แล้วแต่ลอจิกและความต้องการของแต่ละคน

นอนไคล์เอ็นต์แอเรียเม้าส์เมสเสจ

โดยปกติแล้วเราไม่ค่อยจะยุ่งกับเมสเสจกลุ่มนี้ มักจะปล่อยให้ DefWindowProc จัดการแทนเนื่องจากเมสเสจเหล่านี้มีผลสืบเนื่องต่อระบบ แต่อย่างไรก็ดีเมสเสจที่เกิดและจัดอยู่ในกลุ่มนี้มีจำนวน 10 เมสเสจ เท่ากับที่มีในกลุ่มข้างต้น และชื่อที่คล้ายกันด้วยเพียงแต่ในชื่อมีคำ "NC" เติมหน้าแสดงถึงว่าเป็นเมสเสจที่เกิดในส่วนนอนไคล์เอ็นต์แอเรียดังนี้

WM__NCLBUTTONDOWN
เมสเสจแสดงถึงการกดปุ่มซ้ายของเม้าส์

WM__NCLBUTTONUP



เมสเสจแสดงถึงการปล่อยปุ่มซ้ายของเม้าส์

WM_NCLBUTTONDBLCLK

เมสเสจแสดงถึงการเกิดดับเบิ้ลคลิ๊กบนปุ่มซ้ายของเม้าส์

WM_NCMBUTTONDOWN

เมสเสจแสดงถึงการกดปุ่มกลางของเม้าส์ (กรณีเม้าส์ 3 ปุ่ม)

WM_NCMBUTTONUP

เมสเสจแสดงถึงการปล่อยปุ่มกลางของเม้าส์

WM_NCMBUTTONDBLCLK

เมสเสจแสดงถึงการเกิดดับเบิ้ลคลิ๊กบนปุ่มกลางของเม้าส์

WM_NCRBUTTONDOWN

เมสเสจแสดงถึงการกดปุ่มขวาของเม้าส์

WM_NCRBUTTONUP

เมสเสจแสดงถึงการปล่อยปุ่มขวาของเม้าส์

WM_NCRBUTTONBLCLK

เมสเสจแสดงถึงการเกิดดับเบิ้ลคลิ๊กบนปุ่มขวาของเม้าส์

WM_NCMOUSEMOVE

เมสเสจแสดงถึงการเคลื่อนเม้าส์

แต่อย่างไรก็ตามพารามิเตอร์ wParam และ lParam สำหรับนอนไคล์เอ็นต์แอเรียแตกต่างจากพารามิเตอร์ของเมสเสจกลุ่มไคล์เอ็นต์แอเรียโดยที่พารามิเตอร์ wParam จะเก็บค่ารหัส Hit-Test ซึ่งบอกส่วนที่เม้าส์อยู่ในขณะนั้น ทั้งนี้ค่าของรหัส Hit-Test ถูกกำหนดไว้ใน WINDOWS.H (โดยรหัส Hit-Test จะขึ้นต้นด้วยคำว่า "HT") ตัวอย่างเช่น เมื่อเราเลื่อนเม้าส์อยู่ในส่วนหัว (Caption) ของหน้าต่าง เมสเสจ WM_NCMOUSEMOVE เกิดขึ้น และ wParam จะมีค่าเท่ากับรหัส Hit-Test คือ HTCAPTION เป็นต้น

```
POINT *lpPoint;
...
ScreenToClient (hWnd, lpPoint);
/* เปลี่ยนตำแหน่งที่เทียบจากหน้าจอไปเป็นตำแหน่งที่เทียบจากส่วนของไคล์เอ็นต์แอเรีย */
ClientToScreen (hWnd, lpPoint);
/* เปลี่ยนตำแหน่งที่เทียบภายในไคล์เอ็นต์แอเรียไปเป็นตำแหน่งที่เทียบจากหน้าจอ */
```

รูปที่ 4

```
typedef struct tagPOINT {
    int x;
    int y;
} POINT;
```

รูปที่ 5

```
WM_NCHITTEST (Hit-Test Mouse Message) &
WM_MOUSEACTIVATE (Mouse-Activate Message) &
WM_SETCURSOR (Set-Cursor Message)
```

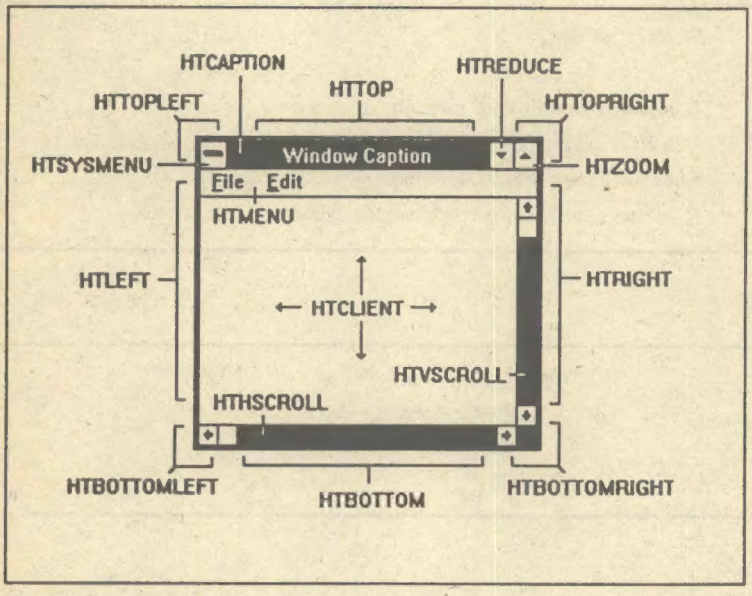
รูปที่ 5.1

พารามิเตอร์ lParam ที่มากับเมสเสจกลุ่มนี้จะเก็บค่าตำแหน่งของเม้าส์เทียบตำแหน่งมาจากตำแหน่งของหน้าจอ (Screen Coordinate) ซึ่งต่างจากตำแหน่งที่เทียบจากส่วนไคล์เอ็นต์แอเรีย (ดูรูปที่ 2 ประกอบ) อย่างไรก็ตามเราสามารถเปลี่ยนตำแหน่งทั้งสองไปมากันได้โดยใช้ฟังก์ชัน ดังรูปที่ 4

พารามิเตอร์ lpPoint เป็น พอยน์เตอร์ชี้ไปยัง POINT ซึ่ง POINT ถูกกำหนดไว้ใน WINDOWS.H ดังรูปที่ 5

เมสเสจ WM_NCHITTEST และ WM_MOUSEACTIVATE และ WM_SETCURSOR เป็นเมสเสจชุดสุดท้ายที่จะแนะนำ โดยทั่วไปมักปล่อยให้กับฟังก์ชัน DefWindowProc จัดการแทน ในรูปที่ 5.1

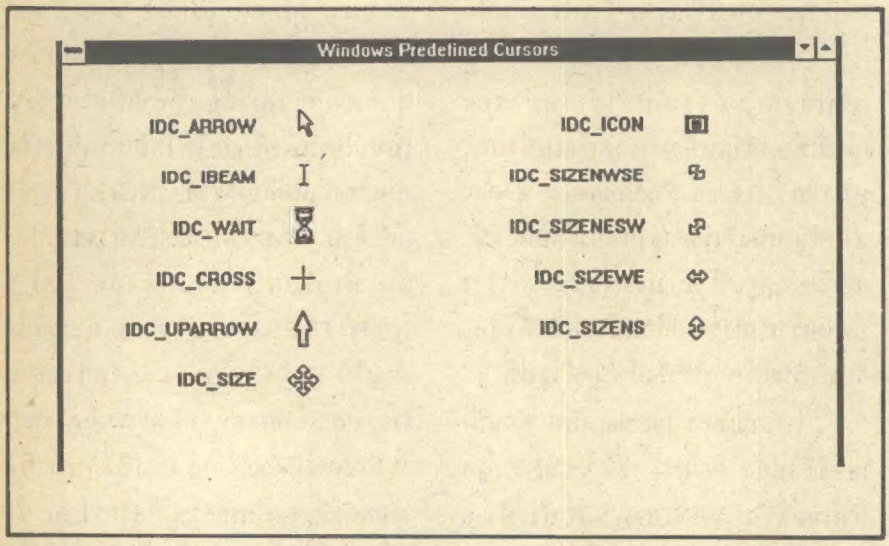
WM_NCHITTEST เกิดทุกครั้งเมื่อเม้าส์ของคุณเคลื่อนที่หรือปุ่มของเม้าส์ถูกคลิ๊ก (click) ซึ่งระบบจะนำ WM_NCHITTEST ไปสร้างเมสเสจเกี่ยวเม้าส์อื่นที่สืบเนื่อง เช่น เมื่อเลื่อนเม้าส์ภายใต้ไคล์เอนต์แอเรีย เมสเสจที่เกิดคือ WM_NCHITTEST แล้วตาม WM_MOUSEMOVE เป็นต้น สำหรับการที่ระบบนำเอา WM_NCHITTEST ไปใช้ก็ต่อเมื่อเราส่งผ่านให้ DefWindowProc จัดการแทนเรา และเมื่อผ่านการจัดการของ DefWindowProc แล้วผลที่ตามมาคือ wParam จะเก็บค่ารหัส Hit-Test ซึ่งจะบอกได้ว่า ขณะนั้นเม้าส์อยู่ส่วนใดของหน้าต่าง ซึ่งเมื่อรู้ว่าเป็นส่วนใดแล้วระบบจึงสามารถสร้างเมสเสจที่แท้จริงออกมาได้ สำหรับ lParam ของ WM_NCHITTEST จะเก็บค่าตำแหน่งแนวแกน x และ y โดยเทียบจากสกรีน (Screen) และ wParam ไม่ใช้ (หมายเหตุ : แสดงค่ารหัส Hit-Test



รูปที่ 6 แสดงรหัส Hit-Test

```
HCURSOR LoadCursor (HANDLE hInstance, LPSTR lpCursorName)
```

รูปที่ 7



รูปที่ 8 แสดงเคอร์เซอร์ที่ระบบจัดเตรียมให้

ตามรูปที่ 6)

WM_MOUSEACTIVATE เป็นเมสเสจที่เกิดขึ้นเมื่อ ปุ่มของเม้าส์ถูกกดและสุดท้ายคือ WM_SETCURSOR เป็นเมสเสจที่เกิดขึ้นทุกครั้งเมื่อเม้าส์มีการเลื่อนเปลี่ยนตำแหน่งหรือปุ่มบนเม้าส์ถูกคลิ๊ก

โดยสรุปแล้วทั้งสามเมสเสจเป็นเมสเสจที่เราควรส่งให้ DefWindowproc จัดการแทนเราจึงจะดีที่สุด ยกเว้นบางประเด็นซึ่งไม่ขอกล่าวต่อในที่นี้

เคอร์เซอร์

เคอร์เซอร์คือรูปหรือบิตแม็พ (Bitmap) ที่เรากำหนดใช้แทนเม้าส์บนสกรีน การใช้เคอร์เซอร์จะมีส่วนทำให้โปรแกรมน่าสนใจมากขึ้นอีก การเลือกใช้เคอร์เซอร์มี 2 วิธีดังนี้

1. เลือกใช้เคอร์เซอร์โดยวิธีลงทะเบียน วิธีนี้เป็นการโยนการดูแลให้กับระบบเพราะเมื่อเลื่อนเม้าส์ไปที่หน้าต่างหนึ่ง ๆ ระบบจะเปลี่ยนรูปเคอร์เซอร์ให้เองตามที่กำหนดลงทะเบียนไว้ เช่น เมื่อคุณเลื่อนเม้าส์ภายใต้โปรแกรม Spy เคอร์เซอร์มีรูปร่างเป็นหัวลูกศร เมื่อเปลี่ยนหน้าต่างเป็นหน้าต่างของโปรแกรม Paintbrush และเลื่อนเม้าส์ภายใต้หน้าต่างนี้ เคอร์เซอร์จะเปลี่ยนเป็นกากบาท เป็นต้น

สำหรับการใช้เคอร์เซอร์โดยปกติโหลดข้ามาก่อนโดยเรียกผ่าน ฟังก์ชัน LoadCursor ดังรูปที่ 7

hInstance คือ ค่าประจำโมดูลหรือโปรแกรมนั้น ซึ่งค่านี้ระบบจะกำหนดให้เอง กรณีใช้เคอร์เซอร์ที่ระบบเตรียมให้ hInstance เท่ากับ NULL

lpCursorName คือ ชื่อที่ใช้แทนเคอร์เซอร์ ที่กำหนดไว้ในไฟล์ rc กรณีไม่ใช้เคอร์เซอร์ของระบบ แต่ถ้าใช้เคอร์เซอร์ที่ระบบเตรียมให้ แทนที่จะส่งชื่อเราจะส่งรหัสของเคอร์เซอร์เข้าไปแทน ดังรูปที่ 8

ค่าที่ได้จากฟังก์ชันนี้ คือ ค่าที่ใช้เป็นตัวแทนเคอร์เซอร์ที่โหลดเข้ามา (สำหรับ HCURSOR กำหนดใน WINDOWS.H มีขนาดเท่ากับเวิร์ด) ดังรูปที่ 9

นตฺถิ ปญฺญาสมา อาภา
สถาบันบัณฑิตพัฒนบริหารศาสตร์

นตฺถิ ปญฺญาสมา อาภา
สถาบันบัณฑิตพัฒนบริหารศาสตร์

นตฺถิ ปญฺญาสมา อาภา
สถาบันบัณฑิตพัฒนบริหารศาสตร์